Project Description
(This is a translation of the original description)

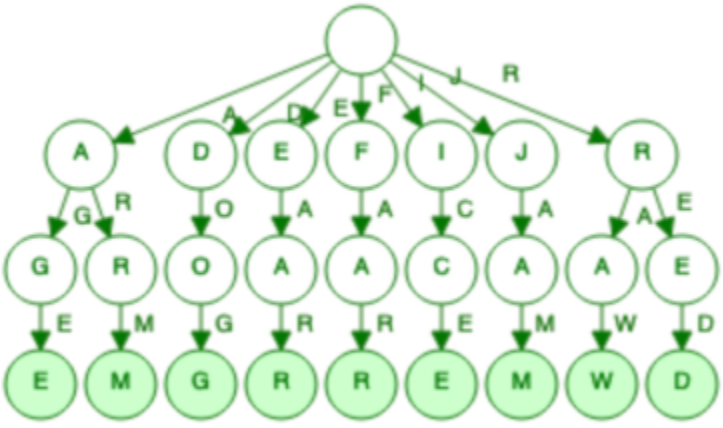
Phase 1:

Goal: Finding a Regex in text using tree structures
In this phase, we were asked to create a search tree from a given text. Then we had to answer some test cases (words) which were a regex including a “\S*”. “\S” matches with any arbitrary character, except whitespaces, and “*” means any number of the prior symbol can be accepted. For instance, “asi\S*” will match "asinine ", "aside ", and “asiatic”.
Choosing the type of tree structure was left to us (I chose “Trie”). We had to take into account the speed and memory of our structure.

Example of given text (After splitting text to words):

ice	ear	jam	dog	far	raw	red	arm	age

Trie representation of the given text:



After creating our model of given text, we are ready to answer queries of the explained format. For instance, if we are asked about words like “\S*r”, we can retrieve "ear " and "far" from our tree. We had to be

careful about the fact that “\S*” wildcard could be anywhere in the query (at the beginning, in the middle, or at the end).

Input and Output format:

- Input:
  In the first line of input we have two numbers, ‘n’ and ‘q’.
  ‘n’ shows the number of words in the given text.
  ‘q’ shows the number of queries.
  In the next line, all of the words of the text are given, space separated. Then, in the next ‘q’ lines, each of the queries is given.
- Output:
  In each line, first we should print the number of words that matched with the respective query, and then print all of the matched words.

Example)

- Input)
  11 3
  mon monk money month monad mongos sidney monster color colour sydney
  mon\S*
  col\S*r
  s\S*dney
- Output)
  7 mon monk money month monad mongos monster
  2 color colour
  2 sidney sydney

Phase 2:

Goal: Finding the text including a specific Regex using tree structures and Hash Tables.

In this phase, initially, some texts are given (with filenames DocIDxx.txt where xx is the id of the text). Then, for each query, we are supposed to return the id of texts which have at least one word that match the query. If there was no doc that matches the pattern, we should return '-1'.
For instance, these three texts are given:

DocID01.txt	DocID02.txt	DocID03.txt
Asdfg	As	Cv
Zxcv	Askew	Luhefhj
Qwer	Asytre	pergh
Poiu	Aspuoiy	
plkm	s	

If the query is "as\S*", then our program must return '1' and '2. Because in the first text, the word "Asdfg", and in the second text, the words: "As", "Askew", "Astyre", and "Aspuoiy" match the pattern.

Note 1: We had to use Hash Tables in this phase.
Note 2: Words in texts may have been hyphenated, uppercase, or lowercase. In order to unify all of the words, we had to omit the hyphens, and make all of the letters of words lowercase. For instance, Long-Term -> longterm.
Note 3: When reporting the docs which included the query, we had to order them by the number of occurrences of the respective pattern in the docs. For example, if we have two docs that matched a pattern, and the text DocID01.txt had 1 instance of the pattern and the text DocID02.txt had 4 instances of the pattern, then we should report this way:
1 4